



Recueil d'Exercices Corrigés

Python

version 1.2

Décembre 2017



Avant-propos

Pour joindre le compilateur des exercices :

✉ eric.berthomier@free.fr

Sources

- Les illustrations sont extraites du cours de Bob CORDEAU et Laurent POINTAL. Elles ont été réalisées par Hélène CORDEAU.
- Les exercices sont inventés ou piochés au gré de mes clics sur la toile. Je citerai notamment : TiProf : <http://tiprof.fr>

L'ennui ...

L'ennui est entré dans le monde par la paresse. - Jean de La Bruyère -

Aussi, afin d'éviter cette paresse, une fois fini l'ensemble des exercices, il est possible d'aller jouer aux Mathématiques Informatiques, en se rendant sur le site de mon cher ami Laurent Signac :

<https://deptinfo-ensip.univ-poitiers.fr/pydefis/>

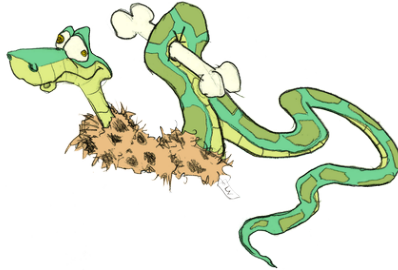
Versions

Ce recueil évolue en fonction des demandes et de mon temps ... toute contribution est la bienvenue.

Décembre 2014	Version initiale	1.0
Janvier 2016	Ajout d'exercices sur la récurrence les expressions régulières	1.1
Septembre 2017	Ajout d'exercice CRobot (POO) introduction du mot clé super()	1.2



Mise en bouche



Rappel des principales commandes



Typeage	type ()
Affectation	=
Saisie	input ()
Commentaires	# Ceci est un commentaire

Avant de commencer...

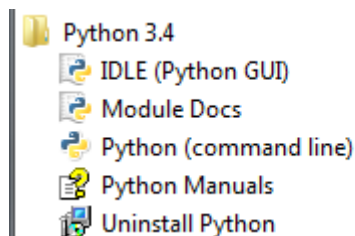
La réalisation de ces travaux pratiques nécessite l'installation de Python 3 et pour plus de commodités de Notepad++ (Windows) ou Scite (Linux).

- Python : <https://www.python.org/downloads/>
- Notepad++ : <http://notepad-plus-plus.org/fr/>
- Scite : <http://www.scintilla.org/SciTEDownload.html>

Langage Python (Windows)

L'interpréteur est lancé par l'item correspondant ajouté dans le menu Démarrer à l'issue de l'installation.

Par exemple :



À l'aide de NotePad++, saisir le script suivant et l'enregistrer sous le nom `hellow.py` :

```
#! C:\Python34\python
# -*- coding: UTF-8 -*- pour spécifier le codage des caractères

print (5) # affichage de la valeur 5

a = 5
print (a) # affichage du contenu de a
print ("a = ", a) # affichage d'un message et du contenu de a
```

```

b = 5.50
print ("b = ", b) # affichage d'un message, du contenu de a et de b sur la même ligne

c = 5,50,14
print ("c = ",c)

texte="Mon texte"
print (texte)
print()

nom = input("Quel est votre nom : ")
print ("nom = ",nom)

# Affichage des types
print ()
print ("type de a: ",type(a))
print ("type de b: ",type(b))
print ("type de c: ",type(c))
print ("type de texte: ",type(texte))
print ()
print ("type de nom: ",type(nom))

```

Le chemin de l'exécutable Python est nécessaire pour que le script puisse désigner l'interpréteur qui doit exécuter ce dernier.

Remarque

✓ Une fois sauvé avec l'extension .py, le script s'exécute quand on clique sur son icône.

Il est aussi possible d'exécuter le script en ligne de commande :

```
.\hellow.py
```

Langage Python (Linux)

L'interpréteur est lancé en ligne de commande par l'exécution de :

```
python
```

Cette commande peut être modifiée en fonction de la version, que l'on veut exécuter.

À l'aide de votre éditeur favori, saisir le script suivant et l'enregistrer en hello.py :

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-# pour spécifier le codage des caractères

print (5) # affichage de la valeur 5

a = 5
print (a) # affichage du contenu de a
print ("a = ", a) # affichage d'un message et du contenu de a

b = 5.50
print ("b = ", b) # affichage d'un message, et du contenu de b

c = 5,50,13
print ("c = ",c)

texte="Mon texte"
print (texte)
print()

nom = input("Quel est votre nom : ")
print ("nom = ",nom)

# Affichage des types
print ()
print ("type de a: ",type(a))
print ("type de b: ",type(b))
print ("type de c: ",type(c))
print ("type de texte: ",type(texte))
print ()
print ("type de nom: ",type(nom))

```

Le chemin de l'exécutable Python est nécessaire pour que le script puisse désigner l'interpréteur, qui est capable de l'exécuter.

Pour pouvoir exécuter le script, il faut en premier lieu rendre ce dernier exécutable.

```
chmod +x hello1.py
./hello1.py
```

Application directe du cours

1. Écrire un programme, qui définit 3 variables : une variable de type texte, une variable de type nombre entier, une variable de type nombre décimal et qui affiche leur type.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

chaine="ceci est une chaîne"
entier=12
decimal=15.0

print (type(chaine))
print (type(entier))
print (type(decimal))
```

2. Affecter dans une même ligne les 3 variables précédemment définies.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

chaine,entier,decimal=("ceci est une chaîne",12,15.0)

print (chaine, type(chaine))
print (entier, type(entier))
print (decimal, type(decimal))
```

3. Cet exercice est sans rapport avec le précédent ...

Écrire un programme qui, à partir de la saisie d'un rayon et d'une hauteur, calcule le volume d'un cône droit : $V = \frac{1}{3} \times \pi \times r^2 \times h$.

Comparer la précision de calcul avec votre calculatrice ou celle de l'ordinateur.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

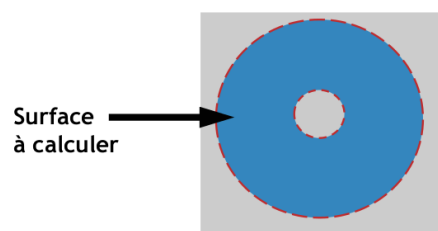
from math import *

h=float(input("Hauteur : "))
r=float(input("Rayon : "))

V=1/3 * pi * r**2 * h

print ("Volume : ", V)
```

4. Une machine découpe dans une plaque, des disques circulaires de rayon r_{Ext} , percés d'un trou circulaire de rayon r_{Int} avec $r_{Int} < r_{Ext}$ et ne débordant pas du disque. Quelle est la surface d'un disque découpé ?



Guide :

- (a) trouver les données
- (b) effectuer les calculs
- (c) afficher le résultat

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from math import *

rayon_Ext = float(input("Rayon extérieur : "))
rayon_Int = float(input("Rayon intérieur : "))

sGrandDisque = pi * rayon_Ext**2
sDuTrou = pi * rayon_Int**2
surface = sGrandDisque - sDuTrou

print ("Surface du disque : ", surface)
```

Application réfléchie

1. Écrire un programme qui affiche le type du résultat des instructions suivantes :

```
— a=3
— a==3
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

aff1=a=3
aff2=a==3

print (aff1)
print (type (aff1))
print (aff2)
print (type (aff2))
```

2. Écrire un programme, qui ajoute une chaîne de caractères à un nombre entier (Exemple la chaîne "le chat" et le nombre 3 pour donner le chat + 3).

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

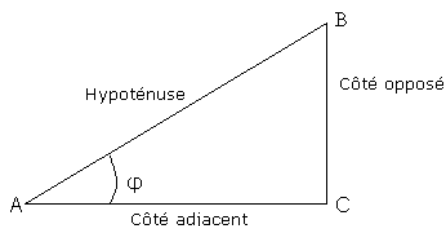
entier = 35
chaine = "Ma chaîne"

somme = str(entier) + chaine

print (somme)
print (type(somme))
```

3. Écrire un programme, qui donne la mesure de l'angle α d'un triangle rectangle, dont on saisit le côté opposé et l'hypoténuse.

Rappel : $\sin \varphi = \text{CoteOppose} / \text{Hypotenuse}$



```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from math import *

opp=float (input("Côté opposé : "))
hyp=float (input("Hypothénuse : "))

print ("angle : ",degrees (asin (opp/hyp)))
```

Application avancée

1. Écrire un programme qui réalise la saisie d'un nombre entier puis affiche la valeur ainsi saisie et son type. Essayer de dépasser la taille maximale des entiers (cf. cours C) avec Python 2 et 3. Expliquer.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

entier = int (input ())

print ("Entier : ",entier)
print ("Type : ",type (entier))
```

Si vous entrez une valeur élevée,

- sous Python 2, l'entier (int) se transformera en long automatiquement (suivant la version de Python).
- sous Python 3, l'entier (int) est conservé. Long n'existe plus

Pour plus d'informations, lire : <http://legacy.python.org/dev/peps/pep-0237/>

2. Lors de la saisie d'un nombre par cast (int (input())) : indiquer une chaîne de caractères en lieu et place d'un nombre, rechercher comment éviter ce bug (aide : commande try)

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

while True:
    try:
        x = int(input("Veuillez saisir un nombre : "))
        break
    except ValueError:
        print("Oops! Ceci n'est pas un nombre. Essayez de nouveau...")

print ("Ok !")
```

Pour plus d'informations, lire : <https://docs.python.org/3/tutorial/errors.html#handling-exceptions>

Culture Générale

1. Quel autre mot courant de l'informatique est issu de la culture Monthy Python ?
SPAM : <http://www.youtube.com/watch?v=anwy2MPT5R>
2. Qu'est-ce qu'un radian ?
Un angle d'1 radian est un angle, qui délimite un arc de cercle d'une longueur égale au rayon du cercle.

Conditions - Itérations - Répétitions



Rappel des principales commandes



Si	if (condition):
Sinon Si	elif (condition):
Sinon	else:
Pour	for variable in
Tant que	while condition :
Longueur d'une chaîne	len (chaîne)

Application directe du cours

1. Écrire un programme `min_max.py`, qui demande de saisir 2 valeurs et qui affiche la plus petite des 2 valeurs.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

valeur1= int(input("Valeur 1 : "))
valeur2= int(input("Valeur 2 : "))

if ( valeur1 < valeur2 ) :
    print ("Valeur la plus petite : %d " % valeur1)
else:
    print ("Valeur la plus petite : %d" % valeur2)
```

2. Écrire un script `longueur_chaine.py`, qui demande de saisir 2 chaînes de caractères et qui affiche la plus grande des 2 chaînes (celle qui a le plus de caractères).

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

chaine1= input("Chaîne 1 : ")
chaine2= input("Chaîne 2 : ")

if ( len(chaine2) > len(chaine1) ) :
    print (("Chaîne la plus grande : " + chaine2 ))
else:
    print (("Chaîne la plus grande : " + chaine1 ))
```

3. Écrire le script `fumeurs.py` vu en cours, en le complétant par des saisies.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

str_fumeur = input ("Fumeur (oui / non) :")
age_fumeur = int (input ("Age du fumeur :"))

if (str_fumeur == "oui"):
    facteur_f = 2
else :
    facteur_f = 0

if (age_fumeur > 60):
    facteur_a = 1
else :
    facteur_a = 0

niveau_de_risque = facteur_f + facteur_a

if niveau_de_risque == 0:
    print ("Le risque est nul !")

if niveau_de_risque != 0:
    print ("Il y a un risque !")

if niveau_de_risque >= 3:
    print ("Risque élevé !")
```

4. Écrire le script `convertir.py`, qui effectue une conversion euros en dollars.

- Le programme commencera par demander à l'utilisateur d'indiquer par un caractère 'E' ou '\$' la devise du montant qu'il va entrer.
- Puis le programme exécutera une action conditionnelle de la forme :

```
if devise == 'E' :
    .....
elif devise == '$' :
    .....
else :
    ..... # affichage d'un message d'erreur
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

devise = input("Devise : ")
montant = int (input ("Montant : "))

# 1 euro = 1.27 $
facteur_euro_dollar = 1.27

if devise == 'E' :
    print ("%f $" % (montant * facteur_euro_dollar))
elif devise == '$' :
    print ("%f Euros" % (montant / facteur_euro_dollar))
else :
    print ("Je n'ai rien compris") # affichage d'un message d'erreur
```

5. Écrire un programme, qui affiche 50 fois "Je dois ranger mon bureau" à l'aide de l'instruction `for`.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

for i in range (1,50):
    print ("Je dois ranger mon bureau")
```

6. Chanson traditionnelle bretonne
La séquence d'instructions

```
n=10
print ("C'est dans %d ans je m'en irai j'entends le loup le renard chanter" % n)
```

permet d'afficher le message :

C'est dans 10 ans je m'en irai j'entends le loup le renard chanter

Écrire une boucle while qui permet d'afficher :

C'est dans 10 ans je m'en irai j'entends le loup le renard chanter

C'est dans 9 ans je m'en irai j'entends le loup le renard chanter

C'est dans 8 ans je m'en irai j'entends le loup le renard chanter

...

C'est dans 1 ans je m'en irai j'entends le loup le renard chanter

Dans un premier temps, on ne s'occupera pas de la faute d'orthographe de la dernière ligne.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

i=10


while ( i > 1 ):
    print ("C'est dans %d ans je m'en irai j'entends le loup le renard chanter" % i)
    i-=1

print ("C'est dans %d an je m'en irai j'entends le loup le renard chanter" % i)
```


Application réfléchie

1. Écrire le script `multiple3.py` qui affiche en fonction d'une valeur saisie l'un des messages suivants :
 - "Ce nombre est pair"
 - "Ce nombre est impair, mais est multiple de 3"
 - "Ce nombre n'est ni pair ni multiple de 3"

Définition

 Un nombre est multiple de 3 si le reste de la division de ce nombre par 3 est égal à 0. Ce reste est nommé modulo

Syntaxe

 `%` permet d'obtenir le modulo en Python

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

nombre = int (input("Nombre : "))

modulo3 = nombre%3
modulo2 = nombre%2

if modulo2 == 0 :
    print ("Ce nombre est pair")
elif modulo3 == 0 :
    print ("Ce nombre est impair, mais est multiple de 3")
else :
    print ("Ce nombre n'est ni pair ni multiple de 3")
```

2. Différence entre `if` et `while`.
On considère les deux programmes suivants :

```
a = 7.5
if a > 3 :
    a = a-1
```

```
a = 7.5
while a > 3 :
    a = a-1
```

Comprendre, calculer et exécuter le programme, afin de donner la valeur de `a`.

3. Écrire un programme qui affiche les nombres de 2 en 2 jusqu'à 100 avec un for puis avec un while.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

for i in range (0,101,2):
    print (i)

i=0

while (i<101):
    print (i)
    i+=2
```

4. Écrire un programme qui affiche les tables de multiplications de 1 à 10.
Aide : utiliser une boucle imbriquée.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

for x in range (1,11):
    for y in range (1,11):
        print ("%d x %d = %d" % (x,y,x*y))
    print ()
```

Application avancée

1. Écrire un programme qui affiche un joli sapin de Noël, dont la taille est donnée par l'utilisateur.
Exemple pour une taille de 12 lignes :

```

      ^
     ^^^
    ^^^^^
   ^^^^^^^
  ^^^^^^^^^
 ^^^^^^^^^^^
^^^^^^^^^^^^^
^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^

```

```
#!/C:\Python34\python
# -*- coding: utf-8 -*-

# Définition du caractère de remplissage
space = " "

# Définition du caractère de dessin du sapin
char_sapin = "^"

# Nombre de lignes constituant le sapin
nbre_ligne = 12

# Nombre de "blancs" à insérer avant le ^
padSize=nbre_ligne - 1

# Position actuelle en lignes dans le dessin du sapin
num_ligne = 0
```

```
# Saut d'un ligne pour ergonomie
print ()

for num_ligne in range (1, nbre_ligne + 1):
    # Nombre de caractères sapin à dessiner
    if ( num_ligne == 1 ):
        nbre_char_sapin = 1
    else:
        nbre_char_sapin = 2 * num_ligne -1

    # Affichage d'une ligne de sapin
    print ( space * padSize, char_sapin * nbre_char_sapin, space * padSize )

    # Décrémenter le nombre de caractères de remplissage
    padSize -= 1
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
import time

def MerryChristmasTree(char,Lines):
    space=' '
    numChars = lambda h : h == 1 and 1 or h + ( h-1 )
    MaxTreeWidth = numChars(Lines)

    padSize = ( Lines -1)
    print
    for lineNumber in range(1,Lines + 1):
        print (space * (padSize), char * numChars(lineNumber),space * (padSize))
        padSize -=1

if __name__ == "__main__":
    # '^' is the character used to build the tree
    # 12 is the height of the tree
    MerryChristmasTree('^',12)
    print ("\nMerry Christmas !")
    time.sleep(5)
```

<http://rollcode.com/print-christmas-tree-python/>

Culture Générale

1. Comment se traduit en Python le `i++`, `i--` du C / C++ en Python ?

```
i+=1
i-=1
```


Utilisation d'un module : Turtle



Rappel des principales commandes

Utiliser Turtle	<code>import turtle</code>
Efface tout	<code>reset()</code>
Aller à	<code>goto (x,y)</code>
Avancer	<code>forward (distance)</code>
Reculer	<code>backward (distance)</code>
Couleur	<code>color (couleur)</code>
Tourner gauche	<code>left (angle)</code>
Tourner droite	<code>right (angle)</code>



Turtle nécessite l'installation de tk-iner : `apt install python-tk` ou `apt install python3-tk`

Application directe du cours

1. Écrire un programme `carre.py` qui trace un carré.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

from turtle import *

for i in range (4) :
    forward(50)
    right(90)

input("Presser entrée pour quitter")
```

2. Écrire un programme, qui trace un triangle équilatéral (3 angles à 60°).

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

from turtle import *

for i in range (3) :
    forward(150)
    right(120)

input("Presser entrée pour quitter")
```

3. Écrire un programme, qui trace un hexagone (polygone à 6 côtés, angles interne à 120°).

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

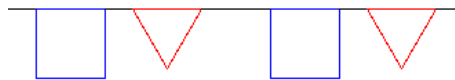
from turtle import *

for i in range (6) :
    forward(50)
    right(60)

input("Presser entrée pour quitter")
```

Application réfléchie

1. Écrire un programme, qui trace un carré, puis un triangle.
Modifier ensuite votre programme pour dessiner n figures consécutives adjacentes.



```
#!/usr/bin/python3
# -*- coding : utf-8 -*-

from turtle import *

setup (1200, 600)

up()
goto (-600,0)
down()

for i in range (8) :

    color('blue')

    for i in range (4) :
        forward(50)
        right(90)

    forward(50)
    color('black')
    forward(20)

    color('red')

    for i in range (3) :
        forward(50)
        right(120)

    forward(50)
    color('black')
    forward(50)

input("Presser entrée pour quitter")
```

2. Écrire un programme, qui trace un carré puis un triangle, qui grossissent au fur et à mesure.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

from turtle import *
# import turtle
```




```

cote_carre=10
cote_triangle=10

setup (1200, 600)

ht()
up()
goto (-600,0)
st()
down()

for j in range (8) :

    color('blue')

    for i in range (4) :
        forward(cote_carre)
        right(90)

    forward(cote_carre)
    color('black')
    forward(20)

    color('red')

    for i in range (3) :
        forward(cote_triangle)
        right(120)

    forward(cote_triangle)
    color('black')
    forward(cote_triangle)

    # On augmente la taille du côté du carré
    cote_carre += 5

    # On augmente la taille du côté du triangle
    cote_triangle += 5

input("Presser entrée pour quitter")

```

Application avancée

1. Écrire un programme, qui trace un cercle (non parfait), sans utiliser la fonction circle de Turtle.

```

#!/usr/bin/python3
# -*- coding :utf-8 -*-

from turtle import *

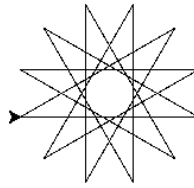
angle = 5
distance = 5
compteur = 0

while compteur <= 72 :
    forward(distance)
    left(angle)
    compteur +=1

input("Presser entrée pour quitter")

```

2. Écrire un programme, qui trace une étoile.



```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

from turtle import *

a = 0

while a < 12 :
    a = a + 1
    forward(150)
    left(150)

input("Presser 0entre pour quitter")
```

3. Étudier le code suivant.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import turtle

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre

tess= turtle.Turtle() # Objet "Turtle"
tess.shape ("turtle") # Une tortue au lieu d'un triangle
tess.color ("green") # Tortue verte

tess.goto(200,200)
tess.right (90)
tess.forward(100)
tess.left (90)
tess.forward(50)

caroline= turtle.Turtle()
caroline.shape ("turtle")
caroline.color ("black")
caroline.forward (30)
caroline.right (90)
caroline.forward(100)
caroline.left (90)
caroline.forward(50)

wn.exitonclick() # On attend un clic sur la croix
```

Culture Générale

1. Turtle est un module de Python, mais s'inspire d'un langage bien plus ancien. Quel est il?

Le Logo :

- [http://fr.wikipedia.org/wiki/Logo_\(langage\)](http://fr.wikipedia.org/wiki/Logo_(langage))
- <http://www.algo.be/logo1/logo-primer-fr.html>

Les Listes



Rappel des principales commandes



Supprimer un élément d'une liste	<code>remove()</code>
Ajouter un élément d'une liste	<code>append()</code>
Trier une liste	<code>sort()</code>
Inverser l'ordre d'une liste	<code>reverse()</code>
Pour	<code>for variable in</code>
Tant que	<code>while condition :</code>
Longueur d'une chaîne	<code>len (chaîne)</code>

Préambule

Dans l'ensemble des ces exercices, la liste suivante sera utilisée :

- lapin
- chat
- chien
- chiot
- dragon
- ornithorynque

Application directe du cours

1. Écrire un programme `liste_animaux.py`, qui initialise la liste et qui affiche l'ensemble des éléments.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

liste_animaux = ["lapin", "chat", "chien", "chiot", "dragon", "ornithorynque"]

for animal in liste_animaux:
    print (animal)
```

- Afficher la liste de manière inversée.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

liste_animaux = ["lapin", "chat", "chien", "chiot", "dragon", "ornithorynque"]
liste_animaux.reverse()

for animal in liste_animaux:
    print (animal)
```

- Afficher la liste de manière triée.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

liste_animaux = ["lapin", "chat", "chien", "chiot", "dragon", "ornithorynque"]
liste_animaux.sort()

for animal in liste_animaux:
    print (animal)
```

- Ajouter (append) l'élément troll dans la liste, puis supprimer l'ensemble des animaux domestiques. Afficher le résultat. Afin de réaliser la suppression, on créera une liste des animaux domestiques.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

liste_animaux = ["lapin", "chat", "chien", "chiot", "dragon", "ornithorynque"]
liste_animaux.append ("troll")

for animal_del in ("lapin", "chat", "chien", "chiot"):
    liste_animaux.remove (animal_del);

for animal in liste_animaux:
    print (animal)
```

Application réfléchie

- Écrire un programme liste_chaine.py, qui donne le nombre de caractères de chaque élément de la liste.

Exemple : lapin possède 5 caractères.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

liste_chaine = ["lapin", "chat", "chien", "chiot", "dragon", "ornithorynque"]

for chaine in liste_chaine:
    print (chaine, "possède", len (chaine), "caractères.")
```

Dans les deux exercices suivants, on considèrera un tableau initialisé avec 10 valeurs aléatoires. Le but des exercices est de dire, si la valeur saisie par l'utilisateur est dans la liste ou non.

Afin de vous aider, voici le début des deux programmes.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import random

tableau_jeu=[]

# Initialisation d'une liste de 10 éléments
for i in range (0,10):
    tableau_jeu.append (random.randint (1,10))
```

2. Recherche séquentielle dans une liste non triée.

Le programme parcourt la liste des valeurs en la comparant une à une à la valeur cherchée, et sort de la boucle :

- soit quand il a parcouru toute la liste sans trouver la valeur
- soit quand la valeur a été trouvée

Si la valeur a été trouvée, le programme annonce "Gagné", sinon, le programme annonce "Perdu".

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import random

tableau_jeu=[]

# Initialisation d'une liste de 10 éléments
for i in range (0,10):
    tableau_jeu.append (random.randint (1,10))

saisie=int (input ("Votre nombre entre 1 et 10 :"))

#~ Recherche séquentielle dans une liste non triée
lg_tableau_jeu = len (tableau_jeu)

# Curseur de position dans le tableau
pos = 0

# On parcourt la liste tant que la valeur n'a pas été trouvée
# ET que l'on ne dépasse pas la taille du tableau
while ((pos < lg_tableau_jeu) and (tableau_jeu [pos] != saisie)):
    pos += 1

if ( pos < lg_tableau_jeu ):
    print ("Gagné")
else:
    print ("Perdu")

print ("\nContrôle visuel")

# On affiche le tirage pour contrôle
for tirage in tableau_jeu:
    print (tirage,end=',')
print()
```

3. Recherche séquentielle dans une liste triée.

Le programme trie les valeurs tirées au sort, puis parcourt la liste des valeurs en la comparant une à une à la valeur cherchée. Il sort de la boucle :

- soit quand il a parcouru toute la liste sans trouver la valeur
- soit quand la valeur lue dans la liste est supérieure à la saisie

Si la valeur a été trouvée, le programme annonce "Gagné", sinon, le programme annonce "Perdu".

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import random

tableau_jeu=[]

# Initialisation d'une liste de 10 éléments
for i in range (0,10):
    tableau_jeu.append (random.randint (1,10))

# On trie le tableau
tableau_jeu.sort ()

saisie=int (input ("Votre nombre entre 1 et 10 :"))

# Curseur de position dans le tableau
pos = 0

# Longueur du tableau
lg_tableau_jeu = len (tableau_jeu)
```

```
# On parcourt la liste tant que la valeur n'a pas été trouvée
while ((pos < lg_tableau_jeu) and ( tableau_jeu [pos] < saisie)):
    pos += 1

if ( tableau_jeu [pos] == saisie ):
    print ("Gagné")
else:
    print ("Perdu")

print ("\nContrôle visuel")

# On affiche le tirage pour contrôle
for tirage in tableau_jeu:
    print (tirage,end=',')
print()
```

4. Jeu de cartes - Une couleur

- Créer un programme jeu_de_cartes.py, qui crée une liste avec l'ensemble des cartes soit : 1,2,...10,V,D,R sans spécifier la couleur. On se permettra une saisie manuelle des valeurs des cartes.
- Contrôler le contenu de votre liste en l'affichant

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

# Initialisation du jeu de cartes
jeu_de_cartes=["1","2","3","4","5","6","7","8","9","10","V","D","R"]

# On affiche le jeu de cartes
for carte in jeu_de_cartes:
    print (carte)

jeu_de_cartes.sort()

print (jeu_de_cartes)
```

5. Jeu de cartes complet Créer une nouvelle liste avec les couleurs, afficher l'ensemble du jeu de cartes. Exemple : R de Coeur.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

# Initialisation du jeu de cartes
jeu_de_cartes=["1","2","3","4","5","6","7","8","9","10","V","D","R"]
tab_couleurs=["Coeur","Carreau","Pique","Trèfle"]

# On affiche le jeu de cartes
for carte in jeu_de_cartes:
    for couleur in tab_couleurs:
        print (( carte + " de " + couleur ))
```

Application avancée

2 Jeux de Cartes

- Créer un programme jeu_de_cartes2.py, qui crée deux listes avec l'ensemble des cartes (soit 2 * 52 cartes).
- Exemple : "1Coeur" pour As de Coeur.
- Contrôler le contenu de vos 2 listes en les affichant
- On tire 10 cartes au hasard dans chaque jeu. Faire afficher les 10 cartes tirées pour chacun des 2 jeux.
- Trouver les cartes identiques dans les 2 tirages.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
# Auteur : Kierian Cousin
```

```

import random

# Déclaration des listes

cartes=["1","2","3","4","5","6","7","8","9","10","V","D","R"]
couleur=["Coeur","Pique","Trefle","Carreau"]

jeudecarte1 = []
jeudecarte2 = []

tirage1 = []
tirage2 = []

# Création des jeux de cartes

for type in couleur :
    for element in cartes :
        combinaison = element + type # Association d'une couleur avec une valeur
        jeudecarte1.append(combinaison) # Ajout de la combinaison à la liste jeudecarte1
        jeudecarte2.append(combinaison)

print(jeudecarte1) # Affichage de la liste jeudecarte1
print(jeudecarte2)

print()

# Choix aléatoire des cartes dans les deux jeux

for i in range(0,10) :
    aleatoire = random.randint(0,len(jeudecarte1)-1) # Choix d'une carte aléatoirement dans la liste jeudecarte1
    tirage1.append(jeudecarte1[aleatoire]) # Ajout de cette carte à la liste tirage1
    aleatoire = random.randint(0,len(jeudecarte2)-1)
    tirage2.append(jeudecarte2[aleatoire])

print(tirage1) # Affichage de la liste tirage1

print()

print(tirage2)

print()

# Indique si deux cartes sont identiques aux deux tirages

for element1 in tirage1 :
    for element2 in tirage2 :
        if element1 == element2 :
            print("%s est identique dans les deux jeux de cartes" %(element1)) # Si elles sont égales, un message s'
            affiche

```

Scrabble©

En compétition de Scrabble©, le tirage des 7 lettres est dit valide quand celui-ci comporte au moins 2 voyelles¹.

Écrire un programme qui tire les lettres dans un sachet constitué des lettres suivantes (simplifié par rapport au vrai jeu) et qui retire les lettres jusqu'à ce que le jeu soit correct.

On ne vous demande qu'un seul tirage valide donc pas de prise en compte des lettres utilisées dans un précédent tirage ...

Instructions : On affichera les tirages incorrects.

- E : 15 lettres
- A : 9 lettres
- I : 6 lettres
- N : 6 lettres
- O : 6 lettres
- R : 6 lettres

1. Cette règle est valable pour les 16 premiers tirages

- S : 6 lettres
- T : 6 lettres

Aide : Utiliser une liste d'éléments et les fonctions qui lui sont associées ...

Corrigé

```
#!/usr/bin/python3
#-*- coding: UTF-8 -*-
import random

x = 0 #Nombre de voyelle dans le tirage.
i = 0 #Incrément pour piocher 7 cartes.
Tab_Scrab = ['E'] * 15 + ['A'] * 15 + ['I'] * 15 + ['N'] * 15 + ['O'] * 15 + ['R'] * 15 + ['S'] * 15 + ['T'] * 15 #
    Notre sac de lettres.
Lettre_piochees = [0] * 7 #Tableau qui recevra les lettres piochées.
Voyelles = ['A', 'E', 'I', 'O', 'U', 'Y'] #Tableau de voyelles pour comparer avec le tirage.

# On mélange le sac de lettres.
random.shuffle(Tab_Scrab)

#on pioche les 7 lettres.
while i < 7 :
    piochex1 = random.choice(Tab_Scrab)
    Lettre_piochees[i] = piochex1
    i = i + 1

#On compare les lettres piochées avec le tableau de voyelle et on incrémente x pour compter le nombre de voyelles dans
    le tirage.
for lettre in Lettre_piochees :
    if lettre in Voyelles :
        x = x + 1

#On affiche les lettres piochées et le nombre de voyelle dans le tirage
print ()
print ("Lettres piochées : ", Lettre_piochees)
print ()
print ("Nombre de voyelles : ", x)

#On vérifie que le nombre de voyelle (x) est superieur.
if x < 2 :
    print ()
    print ( 'ce tirage n\'est pas valide')
    print ()
elif x >= 2 :
    print ()
    print ('ce tirage est valide')
    print ()
```

Faux ami

Problématique de la copie

La copie d'une liste par égal peut faire penser que la copie et son original sont deux éléments dissociés. Ce n'est hélas pas le cas.

```
#!/usr/bin/python

a=[1,2,3]
b=a
b.append (4)
print ("a: ", a)
print ("b: ", b)
```

```
('a: ', [1, 2, 3, 4])
('b: ', [1, 2, 3, 4])
```


Pour réaliser une copie différenciée de l'originale, il faut soit utiliser le module `copy` soit faire la copie à la main.

Copie manuelle

```
#!/usr/bin/python
```

```
a=[1,2,3]
b=[]
for i in a:
    b.append(i)
b.append (4)
print ("a: ", a)
print ("b: ", b)
```

```
('a: ', [1, 2, 3])
('b: ', [1, 2, 3, 4])
```

Module copy

```
#!/usr/bin/python
```

```
import copy
```

```
a=[1,2,3]
b=copy.deepcopy(a)
b.append (4)
print ("a: ", a)
print ("b: ", b)
```

```
('a: ', [1, 2, 3])
('b: ', [1, 2, 3, 4])
```

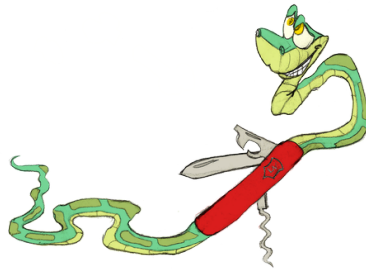
Culture Générale

1. En informatique, qu'est ce qu'un TROLL ?

En argot Internet, un troll est un message (par exemple sur un forum), dont le caractère est susceptible de provoquer des polémiques, ou auquel on ne veut pas répondre et qu'on tente de discréditer en le nommant ainsi. Le mot troll peut également faire référence à un débat conflictuel dans son ensemble, soulevé dans cet objectif.⁰

0. Source Wikipedia : http://fr.wikipedia.org/wiki/Troll_Internet

Fonctions



Rappel des principales commandes



Définition d'une fonction	<code>def nom_fonction (param1, param2, ...):</code>
Retour d'une valeur	<code>return valeur</code>
Vrai	<code>True</code>
Faux	<code>False</code>

Préambule

Dans l'ensemble des exercices ci-dessous, il vous est demandé d'écrire des fonctions ... mais aussi le programme, qui lui est associé afin de les tester !

Application directe du cours

1. `tortue_carre.py` : écrire une fonction `dessine_carre`, qui fait tracer un carré de 50 pixels de côté à la tortue.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from turtle import *

def dessine_carre ():
    for i in range (4):
        forward (50)
        right (90)

dessine_carre()
input()
```

```
#!/usr/bin/python3
# -*- coding :UTF-8 -*-

import turtle

def dessine_carre (tortue):
    for i in range (4):
        tortue.forward (50)
        tortue.right (90)
```

```

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre

tess= turtle.Turtle() # Objet "Turtle"
dessine_carre (tess) # Appel de la fonction carre

wn.exitonclick() # On attend un clic sur la croix

```

2. Modifier le programme précédent de la façon suivante :

- (a) la tortue dessine un carré
- (b) la tortue tourne de 5°
- (c) la tortue avance de 5 pixels
- (d) la tortue dessine un carré

Et ceci 72 fois.

Nommer le tortue_carre_tournant.py

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-
from turtle import *

def dessine_carre ():
    for i in range (4):
        forward (50)
        right (90)
        left (5)
        forward (5)

speed (10)
for i in range (72):
    dessine_carre()

```

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import turtle

def dessine_carre (tortue):
    for i in range (4):
        tortue.forward (50)
        tortue.right (90)
        tortue.left (5)
        tortue.forward (5)

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre

tess= turtle.Turtle() # Objet "Turtle"
tess.speed (10)

for i in range (72):
    dessine_carre(tess) # Appelle la fonction carre

wn.exitonclick() # On attend un clic sur la croix

```

3. Modifier le programme précédent de manière à faire des lunettes ...

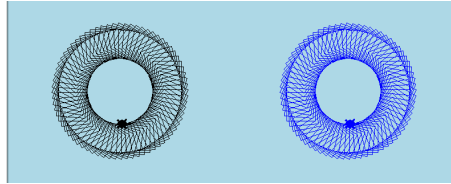
```

#!/usr/bin/python3

import turtle

def dessine_carre (tortue):
    for i in range (4):

```



```

    tortue.forward (50)
    tortue.right (90)

def deplace (tortue):
    tortue.left (5)
    tortue.forward (5)

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen ()
wn.bgcolor ("lightblue")
wn.title ("Python Turtle")

vador = turtle.Turtle()
vador.up()
vador.goto (-200,0)
vador.color("black")
vador.shape("turtle")
vador.speed(5)
vador.down()

yoda = turtle.Turtle()
yoda.up()
yoda.goto (200,0)
yoda.color("blue")
yoda.shape("turtle")
yoda.speed (10)
yoda.down()

for i in range (72):
    dessine_carre(vador) # Appelle la fonction carre
    deplace (vador)
    dessine_carre(yoda) # Appelle la fonction carre
    deplace (yoda)

wn.exitonclick() # On attend un clic sur la croix

```

4. pair_impair.py : écrire 2 fonctions :

- pair (nbre), qui renvoie True, si le nombre est Pair
- impair (nbre), qui renvoie True, si le nombre est Impair

Le nombre sera demandé à l'utilisateur. Le résultat attendu est : "La fonction Pair retourne True pour la valeur 2"

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

def pair (nbre):
    if (nbre % 2 == 0):
        return True

def impair (nbre):
    if (nbre % 2 == 1):
        return True

nombre = int (input ("Votre nombre : "))

if ( pair (nombre) ):
    print ("Le nombre est pair")

if ( impair (nombre) ):
    print ("Le nombre est impair")

```

5. `mini_maxi.py` : écrire 2 fonctions :
- `mini (a,b)` qui renvoie le minimum entre a et b
 - `maxi (a,b)` qui renvoie le maximum entre a et b
- Les 2 nombres a et b seront demandés à l'utilisateur.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

def mini (a,b):
    if ( a < b ):
        return a
    else:
        return b

def maxi (a,b):
    if ( a > b ):
        return a
    else:
        return b

premier_nbre=int (input ("1er nombre : "))
second_nbre=int (input ("2nd nombre : "))

print ("Minimum de %d et %d : %d" % (premier_nbre, second_nbre, mini (premier_nbre, second_nbre)))
print ("Maximum de %d et %d : %d" % (premier_nbre, second_nbre, maxi (premier_nbre, second_nbre)))
```

Application réfléchie

1. Modifier le programme `tortue_carre_tournant.py` en `tortue_carre_couleur.py`, pour que la couleur du carré soit passée en paramètre à la fonction `dessine_carre`.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
from turtle import *

ma_couleur = "red"

def dessine_carre (couleur):
    color (couleur)
    for i in range (4):
        forward (50)
        right (90)
    left (5)
    forward (5)

for i in range (72):
    dessine_carre(ma_couleur)
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import turtle

def dessine_carre (tortue, couleur):
    tortue.color (couleur)
    for i in range (4):
        tortue.forward (50)
        tortue.right (90)
    tortue.left (5)
    tortue.forward (5)

ma_couleur = "red"

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre
```

```
tess= turtle.Turtle() # Objet "Turtle"
tess.shape ("turtle") # Une tortue au lieu d'un triangle
tess.color ("green") # Tortue verte
tess.speed (10)      # Vitesse

for i in range (72):
    dessine_carre(tess, ma_couleur)
```

2. Modifier le programme `tortue_carre_couleur.py` en `tortue_triangle_couleur.py`, pour qu'en lieu et place d'un carré, ce soit un triangle équilatéral, qui soit dessiné.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
from turtle import *

ma_couleur = "green"

def dessine_triangle (couleur):
    color (couleur)
    for i in range (3):
        forward(50)
        right(120)
    left (5)
    forward (5)

for i in range (72):
    dessine_triangle(ma_couleur)
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import turtle

def dessine_triangle (tortue, couleur):
    tortue.color (couleur)
    for i in range (3):
        tortue.forward(50)
        tortue.right(120)
    tortue.left (5)
    tortue.forward (5)

ma_couleur = "red"

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'Ã©cran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenÃªtre
wn.title ("Python Turtle") # DÃ©finit un titre

tess= turtle.Turtle() # Objet "Turtle"
tess.shape ("turtle") # Une tortue au lieu d'un triangle
tess.color ("green") # Tortue verte
tess.speed (10)      # Vitesse

for i in range (72):
    dessine_triangle(tess, ma_couleur)
```

Application avancée

1. `tortue_carre_france.py` : Reprendre le programme `tortue_carre_couleur.py` et créer une liste des trois couleurs du drapeau français, afin de dessiner les carrés avec ces 3 couleurs.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-
from turtle import *

pos_couleur = 0
ma_couleur = ["blue", "white", "red"]
```

```
def dessine_carre (couleur):
    color (couleur)
    for i in range (4):
        forward (50)
        right (90)
    left (5)
    forward (5)

bgcolor ("grey")

for i in range (72):
    dessine_carre(ma_couleur[pos_couleur])
    pos_couleur+=1
    if (pos_couleur == 3):
        pos_couleur=0
```

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import turtle

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("grey") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre

tess= turtle.Turtle() # Objet "Turtle"
tess.shape ("turtle") # Une tortue au lieu d'un triangle
tess.color ("green") # Tortue verte

pos_couleur = 0
ma_couleur = ["blue","white","red"]

def dessine_carre (tortue, couleur):
    tortue.color (couleur)
    for i in range (4):
        tortue.forward (50)
        tortue.right (90)
    tortue.left (5)
    tortue.forward (5)

for i in range (72):
    dessine_carre(tess, ma_couleur[pos_couleur])
    pos_couleur+=1
    if (pos_couleur == 3):
        pos_couleur=0
```

2. `palyndrome.py` : Créer une fonction qui indique, si une chaîne de caractères est ou non un palyn-drome⁰.

Aide

👉 La commande `list(chaîne)` permet de transformer une chaîne de caractères en liste de caractères.

La commande `"".join(list)` permet de transformer une liste en chaîne de caractères.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

def palindrome (mot):

    mot_liste = list (mot)

    mot_liste_reverse = mot_liste
    mot_liste_reverse.reverse ()
    mot_inverse = "".join(mot_liste_reverse)

    if ( mot == mot_inverse ):
        return (True)
```

0. Un palyndrome est un mot qui peut s'écrire dans les deux sens, exemple : radar


```

    else:
        return (False)

saisie = input ("Votre mot : ")

if (palindrome (saisie)):
    print ("Ceci est un palindrome")
else:
    print ("Ceci n'est pas un palindrome")

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

##### Définition de la fonction palindrome #####

def palindrome (mot_test) :
    tab=list(mot_test)
    taille=len(mot_test)

    for i in range (taille) :      #compare le premier caractère avec le dernier, puis le deuxième avec l'avant
        dernier et ainsi de suite
        if tab[i]!=tab[taille-i-1] :      #Si les caractères testés ne sont pas égaux, la fonction palindrome
            retourne "false"
            return (False)

    return (True)

##### Demande de la saisie d'un mot à tester #####

mot = input("Veuillez entrez un mot : ")

##### Affichage du résultat #####

if ( palindrome(mot) == False ) :
    print (mot, " n'est pas un palindrome.")
else :
    print (mot, " est un palindrome")

```

3. pendu.py : Créer un programme de Pendu.

Ce corrigé utilise un dictionnaire de mots mais peut très facilement être adapté.

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

# Auteurs : Kierian Cousin & Eric Berthomier

import random

# Fonction permettant l'importation d'un fichier et son traitement (suppression des retour chariot)
def traitement_fichier () :
    temp=[]                # Déclaration d'une liste
    f = open("dico8.txt",'r') # Assignment d'un fichier à la variable f
    lesLignes = f.readlines() # La variable prend l'ensemble des lignes du fichier
    for i in lesLignes :    # Boucle sur les lignes
        temp.append(i.replace("\n","")) # Chaque ligne s'ajoute à la liste temp
    return (temp)

def affiche_mot_a_trouver (mot_liste):
    print ("Votre jeu : ")
    print ("\t" + " ".join(mot_liste))

liste_mots=traitement_fichier()                # Appel de la fonction traitement_fichier()

mot = liste_mots[random.randint(0,len(liste_mots)-1)]    # Choix d'un mot au hasard dans la liste chaîne

user_word=[]                # Mot trouvé par l'utilisateur
find_word=[]                # Mot à trouver

# Initialisation à rien
for i in range (len(mot)):
    user_word.append("_")    # Ajout de i "_" à la chaîne 1. i correspond au nombre de caractères du mot

```

```

find_word = list(mot)          # Le mot est transposé en liste

max_essais = 11
essais = 1
trouve = False

# Affiche le mot à trouver
affiche_mot_a_trouver (user_word)

# Tant que le nombre d'essais n'est pas écoulé et
# que le mot n'est pas trouvé alors on joue
while ((essais < 11) and ( not trouve)):

    print ()
    print ("Essai n° : %d" % essais)

    caractere = input("Veuillez entrer une lettre : ")

    lettre_trouvee = False

    for element in range (len(find_word)) : # Boucle sur le nombre d'élément de la liste l_mot
        if (find_word[element] == caractere) : # Si l'élément de la liste est égal au caractère saisi
            user_word[element]=caractere      # on remplace dans chaîne l' "_" par le caractère
            lettre_trouvee = True

    if (not lettre_trouvee):
        essais += 1                          # On augmente le nombre d'essais uniquement si la personne n'a pas
            trouvé de lettre

    if (user_word == find_word) :             # Si le mot à trouver est égal à la liste l_mot, c'est gagné
        trouve=True
    else:
        affiche_mot_a_trouver (user_word)

if trouve :
    print ("Vous avez gagné, le mot était %s" % (mot))    # Gagné
else:
    print ("Vous avez perdu, le mot était %s" % (mot))    # Si c'est différent, c'est perdu

```

Culture Générale

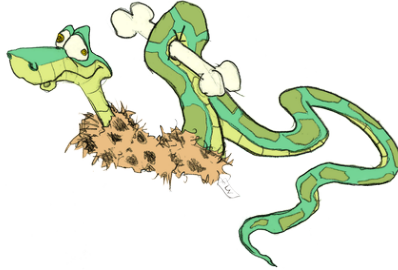
1. Qu'est ce qui différencie une procédure d'une fonction ? Citer un langage de programmation, qui différencie ces deux termes dans sa syntaxe.

Source : <http://www.enib.fr/~tisseau/pdf/course/fonctions-1Paper.pdf>

- Fonction : bloc d'instructions nommé et paramétré, réalisant une certaine tâche. Elle admet zéro, un ou plusieurs paramètres et renvoie toujours un résultat.
- Procédure : bloc d'instructions nommé et paramétré, réalisant une certaine tâche. Elle admet zéro, un ou plusieurs paramètres et ne renvoie pas de résultat.

Exemple : Turbo Pascal

Fichiers et module OS



Rappel des principales commandes



Ouvrir un fichier	open
Fermer un fichier	close
Lire dans un fichier	read / readline
Écrire dans un fichier	write

Application directe du cours

1. Écrire un programme `lire_fichier.py`, qui lit le fichier `loremipsum.txt` et l'affiche.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

f=open("loremipsum.txt")

lignes=f.readlines()
print (" ".join(lignes))

f.close()
```

2. `temperature.py` : Créer un fichier avec un mot par ligne à partir de la liste suivante :
"chaud", "froid", "tempéré", "glacial", "brûlant".

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

f=open("temperature.txt", "w")

liste_temp=["chaud", "froid", "tempéré", "glacial", "brûlant"]

for t in liste_temp:
    f.write (t + "\n")

f.close()
```

3. `temperature_add.py` : Ajouter à ce fichier les équivalents anglais :
"hot", "cold", "moderate", "icy", "ardent"

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

f=open("temperature.txt", "a+")
```

```
liste_temp=["hot","cold","moderate","icy","ardent"]

for t in liste_temp:
    f.write (t+"\n")

f.close()
```

4. Lister le contenu du répertoire courant.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import os

os.system ("dir")
```

Application réfléchie

1. dico.py : Lire le fichier dico.txt et en ressortir les palindromes.

Aide

👉 `mot = mot.replace("\n", "")` permet de supprimer le retour à la ligne d'une chaîne.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

def palindrome (mot):

    mot_liste = list (mot)

    mot_liste_reverse = mot_liste
    mot_liste_reverse.reverse ()
    mot_inverse = "".join(mot_liste_reverse)

    if ( mot == mot_inverse ):
        return (True)
    else:
        return (False)

f=open("dico8.txt")

lignes=f.readlines()

for mot in lignes:
    mot = mot.replace("\n", "")

    if (palindrome (mot)):
        print (mot)

f.close()
```

Culture Générale

1. Qu'est ce que Lorem Ipsum ?

Source : <http://fr.lipsum.com/>

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte. Il n'a pas fait que survivre cinq siècles, mais s'est aussi adapté à la bureautique informatique, sans que son contenu n'en soit modifié. Il a été popularisé dans les années 1960 grâce

à la vente de feuilles Letraset contenant des passages du Lorem Ipsum, et, plus récemment, par son inclusion dans des applications de mise en page de texte, comme Aldus PageMaker

2. En terme SSI, quel peut être l'intérêt de la lecture d'un dictionnaire ?

Lorsque l'on veut mesurer la force d'un mot de passe, on teste les mots du dictionnaire

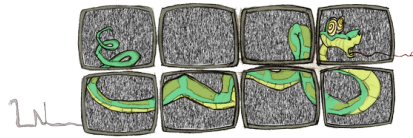
- tels que
- en changeant la casse des caractères
- en insérant des caractères
- en modifiant des caractères par des équivalents

Exemples :

- eric → &RIC@
- 0 → Ø, 1 → I

Plus d'informations : http://fr.wikipedia.org/wiki/Attaque_par_dictionnaire

Programmation Orientée Objet



Astuce sur les modules

Il est possible de tester les modules écrits directement en utilisant `if __name__ == "__main__":`.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import math

def Aire_Cercle (rayon):
    return (math.pi * rayon * rayon )

def Perimetre_Cercle (rayon):
    return (2 * math.pi * rayon)

if __name__ == "__main__":
    r = int (input ("Votre rayon : "))
    print ("Aire : %f" % (Aire_Cercle (r)))
    print ("Perimetre : %f" % (Perimetre_Cercle (r)))
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from module import *

r = int (input ("Votre rayon : "))
print ("Aire : %f" % (Aire_Cercle (r)))
print ("Perimetre : %f" % (Perimetre_Cercle (r)))
```

Application directe du cours

Attention



Aucune création ou héritage de classe n'est demandé pour les exercices de l' "Application directe du cours"

1. `dir_tortue.py` : À l'aide de la commande `dir`, lister l'ensemble des propriétés de l'objet `Turtle`.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-
import turtle

tess= turtle.Turtle() # Objet "Turtle"

print (dir (tess))
```

2. 3tortues.py : Écrire un programme qui créer 3 tortues et qui leur fait effectuer des déplacements et des figures identiques, mais en partant d'une position différente.

- Le déplacement sera implémenté sous la forme d'une fonction s'appliquant à un objet de type tortue.
- La figure fera elle aussi l'objet d'une fonction s'appliquant à un objet de type tortue.

Nom de la tortue	Position de départ
angelo	50,50
donatello	100,100
raphael	150,150

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import turtle

# Déplace la tortue en position x,y
def deplacement (tortue,x,y):
    tortue.up()
    tortue.goto (x,y)
    tortue.down ()

# Trace une figure
def trace (tortue):
    tortue.right (90)
    tortue.forward(100)
    tortue.left (90)
    tortue.forward(50)

turtle.setup (800, 600) # Taille du canevas

wn = turtle.Screen () # La fenêtre d'écran de Turtle
wn.bgcolor ("lightblue") # Couleur de fond de la fenêtre
wn.title ("Python Turtle") # Définit un titre

angelo= turtle.Turtle() # Objet "Turtle"
angelo.shape ("turtle") # Une tortue au lieu d'un triangle
angelo.color ("green") # Tortue verte
deplacement (angelo,50,50)
trace (angelo)

donatello= turtle.Turtle() # Objet "Turtle"
donatello.shape ("turtle") # Une tortue au lieu d'un triangle
donatello.color ("red") # Tortue verte
deplacement (donatello,100,100)
trace (donatello)

raphael= turtle.Turtle() # Objet "Turtle"
raphael.shape ("turtle") # Une tortue au lieu d'un triangle
raphael.color ("blue") # Tortue verte
deplacement (raphael,150,150)
trace (raphael)

wn.exitonclick() # On attend l'appui de la X
```

Application réfléchie

1. Créer une classe Cercle (fichier CCercle.py) qui se définit par son rayon.
La classe CCercle aura 2 méthodes, qui permettront de calculer l'aire ($\pi \times r^2$) et le périmètre ($2 \times \pi \times r$).
Écrire un programme cercle.py qui instancie cette classe et utilise ses méthodes.

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-
```



```
import math

class Cercle ():
    def __init__(self, r):
        self.radius = r

    def surface (self):
        return math.pi*self.radius**2

    def perimetre (self):
        return 2*math.pi*self.radius
```

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

import CCercle

unCercle = CCercle.Cercle(3)
print (unCercle.surface())
print (unCercle.perimetre())
```

2. Créer une classe Domino (fichier CDomino.py) qui se définit par la valeur des points sur ce dernier.

La classe CDomino aura 2 méthodes :

- affiche_points qui affiche le domino
- valeur qui donne la somme des points du domino

Écrire un programme domino.py qui instancie cette classe et utilise ses méthodes :

- Créer 2 dominos [2,6] et [4,3] et leur appliquer les méthodes
- Créer une liste de 7 dominos [6,valeur de 1 à 6] qui les affiche puis indique la somme de l'ensemble.



```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

class Domino ():
    def __init__(self, pa, pb):
        self.pa, self.pb = pa, pb

    def affiche_points (self):
        print ("Face A :",self.pa,end=' ')
        print ("Face B :",self.pb)

    def valeur (self):
        return (self.pa + self.pb)
```

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import CDomino

d1 = CDomino.Domino (2,6)
d2 = CDomino.Domino (4,3)

d1.affiche_points ()
d2.affiche_points ()
print ("Total des points : ",d1.valeur () + d2.valeur ())

liste_dominos = []
```

```

for i in range (7):
    liste_dominos.append(CDomino.Domino(6,i))

vt=0
for i in range (7):
    liste_dominos[i].affiche_points()
    vt += liste_dominos[i].valeur()

print ("Valeur totale des points : ", vt)

```

Application avancée

Jeu de Cartes

1. CJeuDeCartes.py : Créer un classe CJeuDeCartes qui crée un jeu de 52 cartes.
 - Une carte sera composée d’une valeur et d’une forme (coeur, carreau, pique, trèfle).
 - Les méthodes suivantes seront implémentées :
 - nom_carte : affiche le nom d’une carte de manière littérale, ex. Valet de Coeur.
 - battre : mélange le jeu de cartes (on utilisera la fonction shuffle du module random.
 - afficher : permet d’afficher le jeu de cartes.
 - tirer : permet d’extraire une carte du jeu.
 - Vous ferez un jeu de test, qui exécutera les points suivants :
 - (a) Créer un jeu de cartes
 - (b) Mélanger le jeu de cartes (battre)
 - (c) Afficher le jeu de cartes mélangé
 - (d) Tirer une à une toutes les cartes et les afficher

```

#!/usr/bin/python3
# -*- coding: utf-8 -*-

from random import shuffle,randint

class Carte():

    def __init__(self, f, c):
        self.figure = f
        self.couleur = c

    def affiche (self):
        s = self.figure + " de " + self.couleur
        return (s)

class JeuDeCartes ():

    def __init__(self):

        couleur = ["Pique", "Coeur", "Carreau", "Trèfle"]
        valeur = ["As", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Valet", "Dame", "Roi"]

        #- Construction de la liste des 52 cartes
        self.cartes=[]

        for coul in couleur:
            for val in valeur:
                self.cartes.append ( Carte(val,coul) )

        return

        #- Mélange des cartes
    def battre(self):
        shuffle (self.cartes)

        # Tirage la première carte de la pile
    def tirer(self):
        if ( len(self.cartes) == 0 ):

```

```

        print ("Plus aucune carte")
        return
    else:
        t = randint (0, len(self.cartes)-1)
        carte = self.cartes[t]
        del (self.cartes[t])
        return (carte)

# Affichage du jeu de carte
def afficher (self):
    for carte in (self.cartes):
        print (carte.affiche())

```

```

#!/usr/bin/python3
# -*- coding: utf-8 -*-

import CJeuDeCartes

jeu = CJeuDeCartes.JeuDeCartes ()
jeu.afficher()

input ("Nous allons mélanger le jeu de cartes")

jeu.battre()
jeu.afficher()

input ("Nous allons tirer les cartes une à une")

for i in range (52):
    c = jeu.tirer ()
    print ("Je viens de tirer la carte : %s" % (c.affiche()))

```

Robot

Cet exercice est adapté d'un devoir surveillé écrit par Mlle Imene Sghaier.

1. CRobot.py : Créer une classe CRobot qui crée un robot défini par
 - Les caractéristiques suivantes
 - Type
 - SN : Numéro de Série
 - Orientation
 - Status (En service, Hors service, En réparation)
 - Les méthodes suivantes
 - Constructeur Robot () ou Robot (type, sn)
 - getType () : retourne le type de robot
 - getSN () : retourne le numéro de série du robot
 - getOrientation () : retourne l'orientation du robot
 - getStatus () : retourne le status du robot
 - setOrientation (...) : définit l'orientation du robot
 - setEtat(...) : définit l'état du robot
 - tourner(...) : tourne le robot d'un 1/4 de tour
 - afficher() : affiche les informations du robot
 - orientation est un attribut de type entier qui désigne l'orientation du robot.
 - 1 : NORD, 2 : EST, 3 : SUD, 4 : OUEST
 - tourner permet de tourner le robot, par défaut vers la gauche
 - afficher permet d'afficher l'état, l'orientation, le numéro de série et le type du robot.

Instancier cette classe sur un tableau de 4 robots en utilisant l'ensemble des fonctions pour au moins l'un d'eux.
2. CRobotMobile.py : Créer un classe CRobotMobile qui
 - hérite de CRobot
 - se caractérise en plus avec les attributs entiers abs et ord qui définissent la position du Robot-Mobile

- possède une méthode `avancer (...)` qui permet d'avancer le Robot selon son orientation
 - si on avance de x vers l'Est, l'abscisse augmente de x
 - si on avance de x vers l'Ouest, l'abscisse diminue de x
 - si on avance de x vers le Nord, l'ordonnée augmente de x
 - si on avance de x vers le Sud, l'ordonnée diminue de x
 - possède une méthode `affichePosition()` qui affiche la position (coordonnées).
- (a) Écrire un constructeur sans argument de la classe `CRobotMobile`
 - (b) Écrire un constructeur à quatre arguments (`type`, `sn`, `abs`, `ord`) de la classe `CRobotMobile`
 - (c) Redéfinissez la méthode `affiche` tout en utilisant celle de la classe mère et la méthode `affichePosition()`
 - (d) Écrire un programme qui teste votre `CRobotMobile` en lui appliquant les actions suivantes :
 - i. Tourner vers l'Est
 - ii. Avancer de 4 vers l'Ouest
 - iii. Avancer de 6 vers le Nord
 - iv. Avancer de 14 vers l'Est
 - v. Reculer de 8 vers le Sud
 - (e) Proposer une amélioration de gestion pour l'orientation du robot ...

```
class Robot (object):

    def __init__ (self, type, sn, orientation=1, etat=False):
        self.type = type
        self.sn = sn
        self.orientation = orientation
        self.etat = etat

    def getType (self):
        return self.type

    def getSN (self):
        return self.sn

    def getOrientation (self):
        return self.orientation

    def getEtat (self):
        return self.etat

    def setOrientation (self, nlle_orientation):
        self.orientation = nlle_orientation

    def setEtat (self, nl_etat):
        self.etat = nl_etat

    # Par défaut tourne vers la gauche
    def tourner (self, sens = 1):
        if (sens == 1):
            if (self.orientation == 4):
                self.orientation = 1
            else:
                self.orientation += 1
        else:
            if (self.orientation == 1):
                self.orientation = 4
            else:
                self.orientation -= 1

    def affiche (self):
        print ( "Numéro de série : %s " % self.getSN () )
        print ( "Etat : %s " % self.getEtat () )
        print ( "Orientation : %s " % self.getOrientation () )
        print ( "Type : %s " % self.getType () )

class RobotMobile (Robot):
```

```

def __init__ (self, origine_x=0, origine_y=0):
    super().__init__ ("Mobile", "coco")
    self.x = origine_x
    self.y = origine_y

def avancer (self, pas):
    # Nord
    if (self.orientation == 1) :
        self.y += pas

    # Est
    if (self.orientation == 2):
        self.x += pas

    # Sud
    if (self.orientation == 3):
        self.y -= pas

    # Ouest
    if (self.orientation == 4):
        self.x -= pas

def affiche_position (self):
    print ("X: ", self.x)
    print ("Y: ", self.y)

def affiche (self):
    super().affiche ()
    self.affiche_position()
    
```

```

#!/usr/bin/python3
# -*- coding: UTF-8 # pour spécifier le codage des caractères

import CRobot

liste_robot = []

for i in range (1,5):
    robot = CRobot.Robot ("virtuel", i)
    liste_robot.append (robot)

for i in range (0,4):
    liste_robot[i].affiche()
    liste_robot[i].setEtat(True)
    liste_robot[i].setOrientation(i+1)

print ("\n---\nModification des paramètres\n---\n")

for i in range (0,4):
    liste_robot[i].affiche()
    
```

```

#!/usr/bin/python3
# -*- coding: UTF-8 # pour spécifier le codage des caractères

import CRobot

direction = {
    "Nord" : 1,
    "Est" : 2,
    "Sud" : 3,
    "Ouest" : 4
}

robot = CRobot.RobotMobile ()
robot.affiche ()

# Orientation à l'Est
robot.setOrientation (direction["Est"])
robot.affiche ()

# Orientation à l'Ouest
robot.setOrientation (direction["Ouest"])
robot.avancer (4)
robot.affiche()
    
```

```
# Orientation au Nord
robot.setOrientation (direction["Nord"])
robot.avancer (6)
robot.affiche()

# Orientation au Est
robot.setOrientation (direction["Est"])
robot.avancer (14)
robot.affiche()

# Orientation au Sud
robot.setOrientation (direction["Sud"])
robot.avancer (-8)
robot.affiche()

#~ liste_robot = []

#~ for i in range (1,5):
    #~ robot = CRobot.Robot ("virtuel", i)
    #~ liste_robot.append (robot)

#~ for i in range (0,4):
    #~ liste_robot[i].affiche()
    #~ liste_robot[i].setEtat(True)
    #~ liste_robot[i].setOrientation(i+1)

#~ print ("\n---\nModification des paramètres\n---\n")

#~ for i in range (0,4):
    #~ liste_robot[i].affiche()
```

Culture Générale

1. Une méthode de modélisation s'appuie sur l'objet ou inversement. Quelle est elle ?

Source : http://fr.wikipedia.org/wiki/UML_informatique

UML (de l'anglais Unified Modeling Language), ou Langage de modélisation unifié, est un langage de modélisation graphique à base de pictogrammes. Il est utilisé en développement logiciel, et en conception orientée objet. UML est couramment utilisé dans les projets logiciels.

Les Expressions Régulières



Lectures

Lors du parcours de certains sites Python sur les expressions régulières notamment, vous trouverez des syntaxes avec un suffixe `r`.

Afin de bien comprendre ce que ce suffixe produit, voici un exemple et son interprétation.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import re

print (r"\tBonjour, \n et \n au revoir")
print ("\n--\n")
print ("\tBonjour, .* \n et \n au revoir")
```

```
\tBonjour, \n et \n au revoir

--

    Bonjour, .* \n et
au revoir
```

Avec `r`

Le caractère `\` est protégé de toute interprétation :

- `\t` reste `\t`
- `\\n` reste `\\n`
- `\n` reste `\n`

Sans `r`

Le caractère `\` n'est pas protégé de toute interprétation :

- `\t` devient une tabulation
- `\\n` devient `\n`
- `\n` devient un retour chariot

Aide

Expressions régulières

Le site <http://pythex.org/> vous aidera dans la création et la correction de vos expressions régulières, n'hésitez pas à l'utiliser.

Saisie

La lecture d'une chaîne de caractères peut entraîner la prise en compte du retour chariot de fin de saisie, il est possible d'utiliser ce petit code pour enlever ce dernier.

Syntaxe

```
 clear = chaine.rstrip('\r\n')
```

La chaîne `clear` contient alors la chaîne `chaine` nettoyée de tout caractères de retour à la ligne.

Exemple

```
#!/usr/bin/python3
# -*- coding :utf-8 -*-

chaine="coucou\n"

print ("Initial : " + chaine)
print ("-----")

clear = chaine.rstrip('\r\n')

print ("Final : " + clear)
print ("-----")
```

```
Initial : coucou
-----
Final : coucou
-----
```

Application directe du cours

1. Un nombre est composés d'une suite de chiffres. Vérifier que la chaîne saisie est bien celle d'un nombre.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import re

chaine = input ("Votre nombre : ") # Demande à l'utilisateur de saisir un nombre

if (re.search(r"^[0-9]+$",chaine)): # Expression régulière "au moins 1 chiffre"
    print ("La chaîne saisie est bien un nombre ...")
else:
    print ("La chaîne saisie n'est pas un nombre ...")
```

2. Une plaque d'immatriculation est composée de 2 lettres majuscules, un tiret ("-"), 3 chiffres, un tiret ("-") et enfin de 2 lettres majuscules. Vérifier que la chaîne saisie est bien celle d'une plaque d'immatriculation.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import re
```



```
saisie = input("Votre immatriculation ? ")
immat = saisie.rstrip('\r\n')

# Test de l'expression régulière et de la chaîne
if re.search(r"^[A-Z]{2}-[0-9]{3}-[A-Z]{2}$", immat):
    print ("\" + immat + "\" est une plaque mineralogique")
else:
    print ("\" + immat + "\" n'est pas une plaque mineralogique")
```

3. Une adresse IPv4 est composée de 4 nombres entre 0 et 255 séparés par des .. Vérifier que la chaîne saisie est bien celle d'une adresse IP.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import re

saisie_ip = input ("Veuillez saisir une adresse ip : ") # Demande à l'utilisateur de saisir une adresse
email

m_obj=re.search(r"^([0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3})$", saisie_ip)

if m_obj : # Test permettant de dire si la chaîne saisie est une adresse mail
    print (m_obj.group(1) + " est bien une adresse IP")
else :
    print (saisie_ip + " n'est pas une adresse IP.")
```

4. Une adresse mail est composée de caractères alphanumériques suivis de @ suivis d'un . et d'un nom de domaine. Vérifier que la chaîne saisie est bien celle d'une adresse mail.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import re

mail = input ("Veuillez saisir une adresse email : ") # Demande à l'utilisateur de saisir une adresse email

m_obj=re.search(r"^(([a-z0-9])*)(@)(([a-z0-9])*\.[a-z]*)$",mail) # Définition d'une expression régulière

if m_obj:
    print ()
    print (mail + " est bien une adresse email. \n")
    print ("Identifiant : " + m_obj.group(1))
    print ()
    print ("Nom de domaine : " + m_obj.group(4)) # Si oui on l'indique à l'utilisateur et on décompose l'
    adresse mail
else :
    print (mail + " n'est pas une adresse email.")
```

5. Identifier les expressions reconnues par les expressions régulières données ci-dessous. Pour se faire, libre à vous de programmer, tester le site ou d'effectuer les tâches manuellement sachant que le mieux est tout de même de comprendre ce que fait chacune de ces expressions.

Cet exercice est extrait du site <http://home.gna.org/unix-initiation/website/node190.html>

Numéro	Chaîne	Numéro	Chaîne
a	abc	b	zzzz xx
c	abcdef	d	123456 7890 abcaziuz
e	yyyy	f	xyz stuv abc
g	xx abcxxxxxxxxxx	h	xAb* 12345
i	xAB* 45678	j	98745 xAB* 23654
k	abc\$!k;	l	567
m	5666777	n	57
o	Suite... du paragraphe	p	Suite... de l'histoire
q	la suite...	r	Suite.. au prochain numero.

Numéro	Expressions Régulières		
1	c\$	2	c\\$
3	^abc	4	abc\$
5	^abc\$	6	^abc.
7	45	8	^56[67]
9	.56[67]	10	x[Aa][Bb]
11	x[^Aa]	12	[Aa][^b]
13	abcd*	14	566?7
15	[r-v]	16	56*7*
17	56+7+	18	56?7?
19	566?7	20	987 789
21	abc[def]	22	*[Aa][Bb].*12.*
23	*12.*[Aa][Bb]	24	*[Aa]b.*12.*12.*[Aa]b.*
25	*([Aa]b.*12.*12.*[Aa]b).*	26	abc[def][m-x]*
27	^[Ss]uite\\.\\.\\.*		

On regroupe l'ensemble des chaînes et des expressions régulières dans 2 fichiers distincts qu'on utilise en boucle.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import re

# Lecture de l'ensemble des expressions régulières
f = open ("regex.txt", "r")
regexs=f.readlines()
f.close()

# Lecture de l'ensemble des chaînes à analyser
f = open ("chaines.txt", "r")
lignes=f.readlines()
f.close()

for phrase in lignes:

    # On supprime le caractère de fin de ligne
    str = phrase.rstrip('\r\n')

    for reg in regexs:

        # On supprime le caractère de fin de ligne de la regex
        regex = reg.rstrip('\r\n')

        # Test de l'expression régulière et de la chaîne
        if re.search (regex,str):
            print ('"' + str + "\" correspond à " + regex)
        #- else:
            #- print ("\t" + '"' + str + "\" ne correspond pas à " + regex)

    #- print ("-----")
```

Application réfléchie

1. Modifier le code suivant pour rechercher le contenu de texte compris entre <Erreur></Erreur> dans le fichier vandamme.txt en ne prenant pas en compte les espaces inutiles.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import re

# Chaîne de caractères sur laquelle on va travailler
S = 'Bugger all down here on earth!'
```

```

print ("\nCas 1\n----\n")

# Expression régulière visant à matcher le texte
patt = re.compile('Bugger all\s*(.*)here on (.*)!')

# Match
mobj = patt.match(S)

# Affichage du résultat dans le cas où un pattern correspond
if mobj:
    print (mobj.group(1))
    print (mobj.group(2))
else:
    print ("Not find")

# Attention match analyse la chaîne à partir de son début

print ("\nCas 2\n----\n")

# Modification de l'expression régulière visant à matcher le texte
patt = re.compile('all\s*(.*)here on (.*)!')

# Match
mobj = patt.match(S)

# Affichage du résultat dans le cas où un pattern correspond
if mobj:
    print (mobj.group(1))
    print (mobj.group(2))
else:
    print ("Not find")

# On doit donc le corriger en

print ("\nCas 3\n----\n")

# Modification de l'expression régulière visant à matcher le texte
patt = re.compile('.*all\s*(.*)here on (.*)!')

# Match
mobj = patt.match(S)

# Affichage du résultat dans le cas où un pattern correspond
if mobj:
    print (mobj.group(1))
    print (mobj.group(2))
else:
    print ("Not find")

```

Exemple, la saisie de la chaîne : <Erreur> Ceci est une erreur </Erreur> retournera Ceci est une erreur.

```

#!/usr/bin/python3
# -*- coding: utf-8 -*-

import re

# Lecture de l'ensemble du texte
f = open ("vandamne.txt", "r")
vds=f.readlines()
f.close()

# Expression régulière visant à matcher le texte
patt = re.compile('.*<Erreur>\s*(.*)\s*</Erreur>')

# Pour chaque phrase
for phrase in vds:

    print (phrase)

    # Match
    result=patt.match(phrase)

    if (result):

```

```
print ("Find : " + result.group(1))
else:
    print ("Not find !")
```

- Dans le cas où nous aurions voulu chercher plusieurs instances dans une même ligne, il aurait fallu utiliser `findall`.
- `.*` saute tous les caractères jusqu'à la dernière instance de <Erreur>, pour éviter cela, il est possible d'utiliser `.*?`.
- De même pour permettre de s'arrêter au premier </Erreur>, il est nécessaire de coder cet arrêt, en empêchant la recherche de cette chaîne dans `.*` en la remplaçant par `.*?` ou en interdisant </Erreur>

Application avancée

1. Analyser le fichier `script.txt` pour n'isoler que les commandes (entre crochets). On conservera les crochets à l'affichage.
Puis, dans un second temps, l'erreur pour chaque commande, c'est à dire le 3ème champs séparé d'un | si le second champs est erreur. **On affichera uniquement les erreurs différentes des autres.**
 - (a) Écrire un programme qui lit le fichier
 - (b) Isoler les différentes syntaxes à analyser
 - (c) Créer l'expression régulière ou les expressions régulières susceptibles de répondre au problème.
 - (d) Tester votre expression sur un échantillon de lignes
 - (e) Intégrer votre expression régulière dans la lecture du fichier

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import re

# Lecture de l'ensemble du texte
f = open ("script.txt", "r")
scr=f.readlines()
f.close()

# Expression régulière visant à matcher le texte
patt = re.compile('[^]*(\.[.]*\))')

# Pour chaque phrase
for phrase in scr:

    # Match
    result=patt.match(phrase)

    if (result):
        print ("Find : " + result.group(1))
    else:
        print ("Not find !")

# Liste des erreurs
lst_erreurs = [];

# Expression régulière visant à matcher le texte
patt = re.compile('.*\|s*error\s*\|(.*)')

# Pour chaque phrase
for phrase in scr:

    # Match
    result=patt.match(phrase)

    if (result):
        print ("Find : " + result.group(1))
        lst_erreurs.append (result.group(1))
    else:
        print ("Not find !")
```

```
print ("Liste triée")

i=1
for erreur in sorted(set(lst_erreurs)):
    print (str(i) + " : " + erreur)
    i=i+1
```



Anchors		Sample Patterns	
^	Start of line +	([A-Za-z0-9-]+)	Letters, numbers and hyphens
\A	Start of string +	(\d{1,2}\d{1,2}\d{4})	Date (e.g. 21/3/2006)
\$	End of line +	([^\s]+(?:\.(jpg gif png)))\.	jpg, gif or png image
\Z	End of string +	(^[1-9]{1}\$ ^[1-4]{1}[0-9]{1}\$ ^[50]\$)	Any number from 1 to 50 inclusive
\b	Word boundary +	(#?([A-Fa-f0-9]){3}([A-Fa-f0-9]){3})?)	Valid hexadecimal colour code
\B	Not word boundary +	((?=[^\s\d])(?=[a-zA-Z])(?=[A-Z]).{8,15})	8 to 15 character string with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
\<	Start of word	(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
\>	End of word	(\</?[^\>]+\>)	HTML Tags
Character Classes		Note These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.	
\c	Control character		
\s	White space		
\S	Not white space		
\d	Digit		
\D	Not digit		
\w	Word		
\W	Not word		
\xhh	Hexadecimal character hh		
\Oxxx	Octal character xxx		
POSIX Character Classes		Quantifiers	Ranges
[[:upper:]]	Upper case letters	*	0 or more +
[[:lower:]]	Lower case letters	*?	0 or more, ungreedy +
[[:alpha:]]	All letters	+	1 or more +
[[:alnum:]]	Digits and letters	+?	1 or more, ungreedy +
[[:digit:]]	Digits	?	0 or 1 +
[[:xdigit:]]	Hexadecimal digits	??	0 or 1, ungreedy +
[[:punct:]]	Punctuation	{3}	Exactly 3 +
[[:blank:]]	Space and tab	{3,}	3 or more +
[[:space:]]	Blank characters	{3,5}	3, 4 or 5 +
[[:cntrl:]]	Control characters	{3,5}?	3, 4 or 5, ungreedy +
[[:graph:]]	Printed characters		
[[:print:]]	Printed characters and spaces		
[[:word:]]	Digits, letters and underscore		
Assertions		Special Characters	
?=	Lookahead assertion +	\	Escape Character +
?!	Negative lookahead +	\n	New line +
?<=	Lookbehind assertion +	\r	Carriage return +
?!= or ?<!	Negative lookbehind +	\t	Tab +
?>	Once-only Subexpression	\v	Vertical tab +
?()	Condition [if then]	\f	Form feed +
?()	Condition [if then else]	\a	Alarm
?#	Comment	[\b]	Backspace
		\e	Escape
		\N{name}	Named Character
Note		String Replacement (Backreferences)	
Items marked + should work in most regular expression implementations.		\$n	nth non-passive group
		\$2	"xyz" in /\^(abc(xyz))\$/
		\$1	"xyz" in /\^(?:abc)(xyz)\$/
		\$`	Before matched string
		\$'	After matched string
		\$+	Last matched string
		\$&	Entire matched string
		\$_	Entire input string
		\$\$	Literal "\$"
Note		Pattern Modifiers	
		g	Global match
		i	Case-insensitive
		m	Multiple lines
		s	Treat string as single line
		x	Allow comments and white space in pattern
		e	Evaluate replacement
		U	Ungreedy pattern
Note		Metacharacters (must be escaped)	
		^	[
		\$	{
		(\
)	
		<	>

La Récursivité



Application directe du cours

1. Écrire une programme qui calcule la factorielle de n .

En mathématiques, la factorielle d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n . La factorielle est notée !.⁰

Exemple : $6! = 6 * 5 * 4 * 3 * 2 * 1$

La factorielle est utilisée pour dénombrer le nombre de permutations possibles de n éléments.

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

def factoriel (n):
    if n==0:
        return 1
    else:
        return (n * factoriel (n-1))

print factoriel (10)
```

2. Écrire une programme qui calcule les n premiers éléments de la suite de fibonacci.

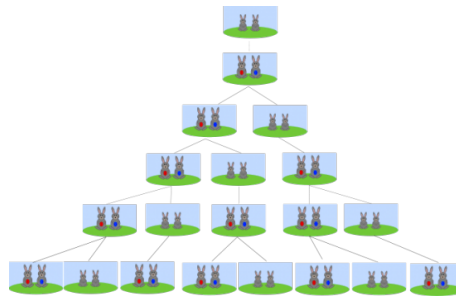
La suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence généralement par les termes 0 et 1 (parfois 1 et 1) et ses premiers termes sont : 0, 1, 1, 2, 3, 5, 8, 13, 21, etc.⁰

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

def fibonacci (n):
    if n==0:
        return 0
    if n==1:
```

0. <https://fr.wikipedia.org/wiki/Factorielle>

0. https://fr.wikipedia.org/wiki/Suite_de_Fibonacci



```

    return 1
    else:
        return fibonacci (n-2) + fibonacci (n-1)

for i in range (0,30):
    print fibonacci (i)

```

Application réfléchie

1. Division Euclidienne.

Tant qu'il nous reste dans a une quantité suffisante pour prendre b , on retranche b de a , c'est-à-dire qu'on prend une fois de plus b de a et donc le quotient augmente d'une unité. Lorsqu'on ne peut plus retrancher b de a (parce que $a < b$) alors le reste de la division euclidienne est a .

```

#!/usr/bin/python2
# -*- coding: UTF-8 -*-

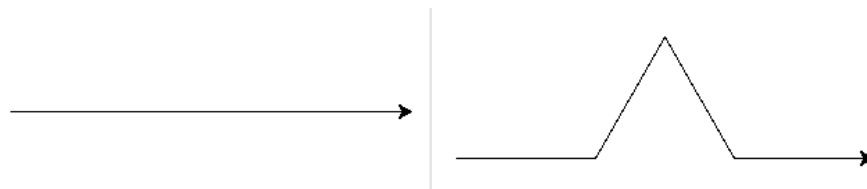
def soustraction (a,b):
    if ( a >= b ):
        return (soustraction (a-b, b))
    else:
        return a

diviseur=int(raw_input("Diviseur : "))
dividende=int(raw_input("Dividende : "))
reste=soustraction(diviseur,dividende)
print ("Reste : " + str (reste))

```

2. Flocon de von Koch.

Le flocon de von Koch s'obtient en partant d'un segment que l'on partage en 3, la partie du milieu est remplacée par un triangle équilatéral comme ceci :



— Écrire une fonction `ligne` qui construit cet élément.

```

#!/usr/bin/python2
# -*- coding: UTF-8 -*-

# programme principal
from turtle import *

def ligne (l):
    forward (l)

```



```
ligne (300)

exitonclick()
```

- Écrire une fonction `triangle` qui construit cet élément (équilatéral).

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

# programme principal
from turtle import *

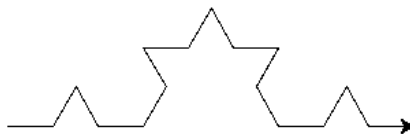
def ligne (l):
    forward (l)

def triangle (l):
    forward (l/3)
    left (60)
    forward (l/3)
    right (120)
    forward (l/3)
    left (60)
    forward (l/3)

triangle (300)

exitonclick()
```

- Remplacer les `forward` de la fonction `triangle` par un tracé de triangle comme défini précédemment, vous devriez obtenir :



```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

# programme principal
from turtle import *

def ligne (l):
    forward (l)

def triangle2 (l):
    forward (l/3)
    left (60)
    forward (l/3)
    right (120)
    forward (l/3)
    left (60)
    forward (l/3)

def triangles (l):
    triangle2 (l/3)
    left (60)
    triangle2 (l/3)
    right (120)
    triangle2 (l/3)
    left (60)
    triangle2 (l/3)

triangles (300)

exitonclick()
```

— Étudier le code suivant.

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

# programme principal
from turtle import *

def segment (l,n):
    if n==0:
        # tracer jusqu'en B
        forward (l)
    else:
        # dessiner un triangle de équilatéral longueur l/3
        segment (l/3,n-1)
        left (60)
        segment (l/3,n-1)
        right (120)
        segment (l/3,n-1)
        left (60)
        segment (l/3,n-1)

def flocon (l,n):
    segment (l,n)
    right(120)
    segment (l,n)
    right(120)
    segment (l,n)

etape=int(raw_input("Donnez le nombre d'étapes (n) : "))
taille=float(raw_input("Donnez la longueur du côté initial (l) : "))

exitonclick()
```

Application avancée

1. Fractale du dragon.

La courbe du dragon se construit ainsi :



- Si $t = 0$, l'ordinateur doit dessiner une ligne. C'est la base (ou l'initiateur). La longueur a peu d'importance. On définit la longueur une fois avec s .
- Sinon, si $t > 0$: $\text{Dragon}(t) = \text{Dragon}(t-1) \neg \text{Dragon}(t-1)$. C'est la règle de récursivité (ou le générateur). L'ordinateur doit dessiner une courbe de dragon avec profondeur de récursion $t-1$. Cela donne :

- Dessiner $\text{Dragon}(t-1)$
- Tourner à gauche (90°)
- Dessiner $\text{Dragon}(t-1)$

Il y a un petit problème : on ne peut pas dessiner $\text{Dragon}(t-1)$ exactement de la même façon les deux fois. En effet, le premier $\text{Dragon}(t-1)$ est dessiné vers l'extérieur en partant du milieu de $\text{Dragon}(t)$. Ensuite on tourne de 90° . Le deuxième $\text{Dragon}(t-1)$ est dessiné à l'inverse du milieu de $\text{Dragon}(t)$ vers l'extérieur. Pour que les deux $\text{Dragon}(t-1)$ soit représentée de la même façon, le deuxième $\text{Dragon}(t-1)$ doit être dessiné en miroir. Cela veut dire que tous les angles (a) sont

en miroir et (b) doivent être dessinés dans l'ordre inverse.

L'astuce consiste à donner un signe qui indique le sens ($vz = 1$ veut dire « + », $vz = -1$ veut dire « - »). On dessine d'abord un Dragon ($t-1$) avec signe positif ($vz = 1$). Ensuite on tourne de 90° et dessinons un Dragon ($t-1$) avec signe négatif ($vz = -1$).

- Dessiner Dragon ($t-1$) signe (+)
- Tourner à gauche ($vz \cdot 90^\circ$)
- Dessiner Dragon ($t-1$) signe (-)

Écrire le programme.

Cette solution est l'adaptation non optimisée de la solution proposée par l'auteur de cet exercice. ⁰

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

# Courbe du dragon avec instructions Logo
from turtle import *
from math import sin, cos, radians

# ----- commandes logo -----
# Définition d'une fonction ligne
# Trace une ligne partant du point (x1,y1) au point (x2,y2)
def ligne (x1, y1, x2, y2):
    up()
    goto(x1,y1)
    down()
    goto(x2,y2)

def fpos(x0,y0):
    # place la tortue en (x0; y0)
    global x,y
    x = x0
    y = y0

def fcap(angle0):
    global angle
    # oriente la tortue dans une direction (en degrés)
    angle = angle0

def av(d):
    # avance en dessinant
    global x, y
    x2 = x + d*cos(angle)
    y2 = y + d*sin(angle)
    ligne(x, y, x2, y2)
    x = x2
    y = y2

def tg(a):
    # tourne à gauche de a degrés
    global angle
    angle -= radians(a)

# -----

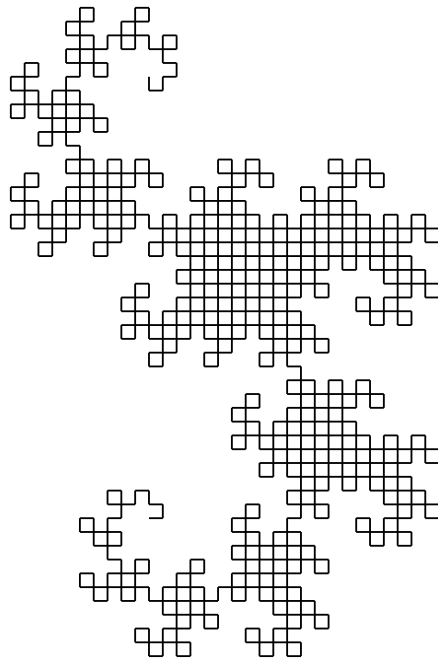
def dragon(t,vz):
    if t==0:
        av(15)
    else:
        dragon(t-1,1)
        tg(vz*90)
        dragon(t-1,-1)

def dessiner():
    fpos(0,0)
    fcap(0)
    dragon(10,1)

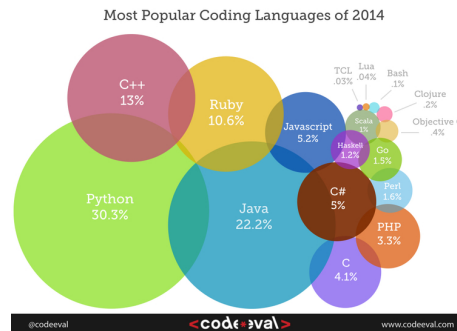
setup (800, 800)
```

0. Je laisse le soin aux élèves de me surprendre en me donnant la correction optimisée

```
dessiner()  
input("Presser entrée pour quitter")
```



Far Away ...



Préambule

Aucun cours n'est associé à ces exercices ...

Les exercices précédent vous ont donné les bases nécessaires pour un apprentissage approfondi du Python, vous trouverez donc des codes à comprendre, étudier et enfin à adapter.

Map / Dictionaries

Cette fonctionnalité appelée aussi Table de Hashage permet d'associer un champs à un autre champs.
Exemple : eric → berthomier

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

favorite_sports = {
    'Ralph Williams' : 'Football',
    'Michael Tippett' : 'Basketball',
    'Edward Elgar' : 'Baseball',
    'Rebecca Clarke' : 'Netball',
    'Ethel Smyth' : 'Badminton',
    'Frank Bridge' : 'Rugby'
}

print(favorite_sports['Rebecca Clarke'])

del favorite_sports['Ethel Smyth']
print(favorite_sports)

favorite_sports['Ralph Williams'] = 'Ice Hockey'
print(favorite_sports)

print ("Sport favori de Rebecca Clarke : ",favorite_sports.get('Rebecca Clarke', "Non trouvé"))
print ("Sport favori de Eric Berthomier : ",favorite_sports.get('Eric Berthomier', "Non trouvé"))

if "Eric Berthomier" in favorite_sports:
    print ("Eric Berthomier est dans la liste")
else:
    print ("Eric Berthomier n'est pas dans la liste")
```

Exercice⁰

- Choisissez 5 mots de la langue française et créez un dictionnaire qui associe à chacun de ces mots sa traduction en anglais.
- Ajoutez une entrée au dictionnaire de la question précédente (un nouveau mot et sa définition).
- Écrivez une fonction `ajoute(mot1, mot2, d)` qui prend en argument un mot en français, sa traduction en anglais et ajoute ces deux mots dans le dictionnaire `d` uniquement si `mot1` n'est pas une clé du dictionnaire.
- Écrivez une fonction qui affiche à l'écran toutes les valeurs correspondant aux clés qui sont dans votre dictionnaire (ici, tous les mots en anglais qui apparaissent dans votre dictionnaire).
Indication : on exécute une boucle `for` sur tous les éléments de `d.keys()` et l'on renvoie pour chacun la valeur qui lui est associée.
- Écrivez une fonction `supprime(car, d)` qui prend en argument un caractère `car` et un dictionnaire `d` et supprime du dictionnaire toutes les entrées correspondant à des clés qui commencent par la lettre `c`.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

def ajoute (mot1 , mot2 , d):
    if not mot1 in d :
        d[mot1] = mot2

def valeurs_sol1 (d):
    for k in d.keys () :
        print (d[k])

def valeurs_sol2 (d):
    for v in d.values () :
        print (v)

def supprime ( car , d ) :
    new_dict = {}
    for k in d.keys () :
        if k[0] != car :
            new_dict [k] = d[k]
    return new_dict

print ("\nDéfinition de la table de hashage\n")
dic = {"chat": "cat", "chien": "dog", "vache": "cow", "tigre": "tiger", "licorne": "unicorn"}
print (dic)

print ("\nAjout de souris\n")
dic ["souris"] = "mouse"
print (dic)

print ("\nAjout de dragon\n")
ajoute ("dragon", "dragon", dic)
print (dic)

print ("\nSecond ajout de dragon\n")
ajoute ("dragon", "smaug", dic)
print (dic)

print ("\nSolution 1\n")
valeurs_sol1 (dic)

print ("\nSolution 2\n")
valeurs_sol2 (dic)

print ("\nSuppression d'éléments\n")
dic=supprime ( "c" , dic )
print (dic)
```

0. Extrait de la page personnelle de Victor Poupet <http://www2.lirmm.fr/~poupet/>

CSV

Ce module permet de lire / créer un fichier CSV (Comma-separated values).

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

import csv

# Transformation du fichier CSV en Dictionnaire
file_dico = csv.DictReader(open("exemple.csv"))

# Affichage du dictionnaire
print (file_dico)

# Affichage du contenu du dictionnaire
for row in file_dico:
    print(row['francais'], row['anglais'])
```

```
francais,anglais
souris,mouse
cameleon,chameleon
chien,dog
vache,cow
epee ,sword
licorne,unicorn
aigle,eagle
dragon,dragon
nain,dwarf
dragon,smaug
noyau,kernel
epee ,stormbringer
tigre,tiger
```

Exercice

Modifier le code précédent pour créer un fichier `c.csv` ne contenant que les mots **français** commençant par `c`. On prendra soin à entourer de guillemets les champs pour les protéger.

```
#!/usr/bin/python2
# -*- coding: UTF-8 -*-

import csv

# Transformation du fichier CSV en Dictionnaire
file_dico = csv.DictReader(open("exemple.csv"))

# Affichage du dictionnaire
print (file_dico)

# Création d'un fichier au format texte ouvert en écriture
f = open("C.csv", 'wt')

# Création de l'entête du fichier CSV
writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
writer.writerow( ('Français', 'Anglais') )

# Affichage du contenu du dictionnaire
for row in file_dico:
    # Découpage du mot en un tableau de caractères
    tab = list(row['francais'])

    # Si le premier caractère est "c" ou "C" on l'écrit dans le fichier
    # Attention au cas de la première ligne si on avait à traiter la lettre f
    if ((tab[0] == "c") or (tab[0] == "C")):
        writer.writerow ( (row['francais'], row['anglais']) )

# Fermeture du fichier
f.close()
```

Tk

Ce module permet de créer des interfaces graphiques à l'aide de composants appelés Widgets.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from tkinter import *

def traceDroite():
    #~ Tracé d'une ligne dans le canevas c
    # coordonnées de la ligne
    fenetre.create_line(10,190,190,10,width=2,fill="green")

#----- Programme principal -----#
#~ Création de la fenetre :
fen = Tk()

# création des widgets (composants) :
fenetre = Canvas(fen,bg='grey',height=200,width=200)

fenetre.pack(side=LEFT)

btn1 = Button(fen,text='Quitter',command=fen.quit)
btn1.pack(side=BOTTOM)

btn2 = Button(fen,text='Tracer une ligne',command=traceDroite)
btn2.pack()

fen.mainloop() # boucle en attente d'événements
fen.destroy() # destruction (fermeture) de la fenêtre
```

Exercice

Modifier le code précédent pour ajouter un bouton qui crée une ligne de taille, de position et de couleur aléatoire.

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from random import randrange
from tkinter import *

def autreCouleur():
    tab_couleur = ("white","black","red","green","blue","cyan","yellow","magenta")

    t = len (tab_couleur)
    couleur = tab_couleur [randrange(t)]

    #~ Coordonnées aléatoires
    x1,x2=randrange(190),randrange(190)
    y1,y2=randrange(190),randrange(190)

    #~ Inversion des coordonnées si besoin
    if (x1>x2) :
        (x1,x2)=(x2,x1)
    if (y1>y2):
        (y1,y2)=(y2,y1)

    fenetre.create_line(x1,y1,x2,y2,width=2,fill=couleur)

def traceDroite():
    #~ Tracé d'une ligne dans le canevas c
    # coordonnées de la ligne
    fenetre.create_line(10,190,190,10,width=2,fill=couleur)

#----- Programme principal -----#
couleur = "green"

#~ Création de la fenetre :
```



```

fen = Tk()

# création des widgets (composants) :
fenetre = Canvas(fen,bg='dark grey',height=200,width=200)

fenetre.pack(side=LEFT)

btn1 = Button(fen,text='Quitter',command=fen.quit)
btn1.pack(side=BOTTOM)

btn2 = Button(fen,text='Tracer une ligne',command=traceDroite)
btn2.pack()

btn3 = Button(fen,text='Random Line',command=autreCouleur)
btn3.pack()

fen.mainloop() # boucle en attente d'événements
fen.destroy() # destruction (fermeture) de la fenêtre

```

PythonPlotting (Non présent en standard)

Ce module permet de créer des graphes.

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import numpy as np
import pylab as pl

# Création d'un tableau de valeur en x
x = [1, 2, 3, 4, 5]

# Création d'un tableau de valeur en y en adéquation avec les x
y = [1, 4, 9, 16, 25]

# Utilisation de pylab pour le tracé en mémoire
pl.plot(x, y)

# Affichage du résultat
pl.show()

```

Exercices

1. Modifier le code précédent pour ajouter une nouvelle courbe.

```

#!/usr/bin/python3
# -*- coding: UTF-8 -*-

import numpy as np
import pylab as pl

# Création d'un tableau de valeur en x
x = [1, 2, 3, 4, 5]

# Création d'un tableau de valeur en y en adéquation avec les x
y = [1, 4, 9, 16, 25]

# Utilisation de pylab pour le tracé en mémoire
pl.plot(x, y)

# Modification du tableau de valeur en y
y = [3, 12, 4, 1, 90]

# Utilisation de pylab pour le tracé en mémoire
pl.plot(x, y)

# Affichage du résultat
pl.show()

```

Culture Générale

1. Un langage de programmation porte le même nom qu'une pierre précieuse, quel est il ?

Source : <https://www.ruby-lang.org/fr/>

Ruby ...est un langage open-source dynamique qui met l'accent sur la simplicité et la productivité. Sa syntaxe élégante en facilite la lecture et l'écriture.